AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A method for deferring execution of instructions with unresolved data dependencies as they are issued for execution in program order, comprising:

issuing instructions for execution in program order during a normal execution mode; and

upon encountering an unresolved data dependency during execution of an instruction,

generating a checkpoint that can subsequently be used to return execution of the program to the point of the instruction, and

executing subsequent instructions in an execute-ahead mode, wherein instructions that cannot be executed because of an unresolved data dependency are deferred, and wherein other non-deferred instructions are executed in program order;

wherein if a non-data-dependent stall condition is encountered while executing in normal mode or execute-ahead mode, the method further comprises:

moving to a scout mode, wherein instructions are speculatively executed to prefetch future loads, but wherein results are not committed to the architectural state of the processor; and

when the launch point stall condition (the unresolved data dependency or the non-data dependent stall condition that originally caused the system to move out of normal execution mode) is finally resolved, using the checkpoint to resume

execution in normal mode from the launch point instruction (the instruction that originally encountered the launch point stall condition).

2. (Original) The method of claim 1, wherein if the unresolved data dependency is resolved during execute-ahead mode, the method further comprises:

executing deferred instructions in a deferred execution mode; and

if all deferred instructions are executed, returning to the normal execution mode to resume normal program execution from the point where the execute-ahead mode left off.

3. (Currently amended) The method of claim 2, wherein executing deferred instructions in the deferred execution mode involves:

issuing deferred instructions for execution in program order;

deferring execution of deferred instructions that still cannot be executed because of unresolved data dependencies; and

executing other deferred instructions that are able to be executed in program order.

4. (Original) The method of claim 3, wherein if some deferred instruction are deferred again, the method further comprises returning to execute-ahead mode at the point where execute-ahead mode left off.

5. (Original) The method of claim 2, wherein generating the checkpoint involves saving a precise architectural state of the processor to facilitate subsequent recovery from exceptions that arise during execute-ahead mode or deferred mode.

6. (Original) The method of claim 1, wherein executing instructions involves keeping track of data dependencies to facilitate determining if an instruction is subject to an unresolved data dependency.

7. (Currently amended) The method of claim 6 ~~claim 1~~, wherein keeping track of data dependencies involves maintaining state information for each register, which indicates whether or not a value in the register depends on an unresolved data-dependency.

8. (Original) The method of claim 1, wherein the unresolved data dependency can include:

a use of an operand that has not returned from a preceding load miss;

a use of an operand that has not returned from a preceding translation lookaside buffer (TLB) miss;

a use of an operand that has not returned from a preceding full or partial read-after-write (RAW) from store buffer operation; and

a use of an operand that depends on another operand that is subject to an unresolved data dependency.

9 (Canceled).

10. (Currently amended) The method of claim 1 ~~claim 9~~, wherein the non-data-dependent stall condition can include:

a memory barrier operation;

a load buffer full condition; and

a store buffer full condition.

11. (Original) The method of claim 1, wherein deferring instructions involves storing instructions in a deferred buffer that is organized as a first-in first-out buffer.

12. (Original) The method of claim 1, wherein issuing instructions for execution in program order involves issuing instructions from an instruction buffer that is organized as a first-in first-out buffer.

13. (Currently amended) An apparatus that defers execution of instructions with unresolved data dependencies as they are issued for execution in program order, comprising:

an execution mechanism configured to issue instructions for execution in program order during a normal execution mode;

a detection mechanism configured to detect an unresolved data dependency;

wherein if an unresolved data dependency is detected during execution of an instruction, the execution mechanism is configured to,

generate a checkpoint that can subsequently be used to return execution of the program to the point of the instruction, and to

execute subsequent instructions in an execute-ahead mode, wherein instructions that cannot be executed because of an unresolved data dependency are deferred, and wherein other non-deferred instructions are executed in program order;

wherein if a non-data-dependent stall condition is encountered while executing in normal mode or execute-ahead mode, the execution mechanism is configured to move into a scout mode;

wherein during scout mode instructions are speculatively executed to prefetch future loads, but results are not committed to the architectural state of the processor; and

wherein during scout mode when the launch point stall condition (the unresolved data dependency or the non-data dependent stall condition that originally caused the system to move out of normal execution mode) is finally resolved, the checkpoint is used to resume execution in normal mode from the launch point instruction (the instruction that originally encountered the launch point stall condition).

14. (Original) The apparatus of claim 13,

wherein if the unresolved data dependency is resolved during execute-ahead mode, the execution mechanism is configured to execute deferred instructions in a deferred execution mode; and

wherein if all deferred instructions are executed during deferred execution mode, the execution mechanism is configured to return to normal execution mode to resume normal program execution from the point where the execute-ahead mode left off.

15. (Currently amended) The apparatus of claim 14, wherein while executing deferred instructions in the deferred execution mode, the execution mechanism is configured to:

issue deferred instructions for execution in program order;

defer execution of deferred instructions that still cannot be executed because of unresolved data dependencies; and to

execute other deferred instructions that are able to be executed in program order.

16. (Original) The apparatus of claim 15, wherein if some deferred instruction are deferred again, the execution mechanism is configured to return to execute-ahead mode at the point where execute-ahead mode left off.

17. (Original) The apparatus of claim 14, wherein while generating the checkpoint, the execution mechanism is configured to save a precise architectural state of the processor to facilitate subsequent recovery from exceptions that arise during execute-ahead mode or deferred mode.

18. (Original) The apparatus of claim 13, wherein the execution mechanism is configured to keep track of data dependencies to facilitate determining if an instruction is subject to an unresolved data dependency.

19. (Currently amended) The apparatus of claim 18 ~~claim 13~~, wherein while keeping track of data dependencies, the execution mechanism is configured to maintaining state information for each register, which indicates whether or not a value in the register depends on an unresolved data-dependency.

20. (Original) The apparatus of claim 13, wherein the unresolved data dependency can include:

a use of an operand that has not returned from a preceding load miss;

a use of an operand that has not returned from a preceding translation lookaside buffer (TLB) miss;

a use of an operand that has not returned from a preceding full or partial read-after-write (RAW) from store buffer operation; and

a use of an operand that depends on another operand that is subject to an unresolved data dependency.

21 (Canceled).

22. (Currently amended) The apparatus of claim 13 ~~claim 21~~, wherein the non-data-dependent stall condition can include:

a memory barrier operation;

a load buffer full condition; and

a store buffer full condition.

23. (Original) The apparatus of claim 13, further comprising a deferred buffer for storing deferred instructions, wherein the deferred buffer is organized as a first-in first-out buffer.

24. (Original) The apparatus of claim 13, further comprising an instruction buffer for storing instructions to be issued for execution in program order, wherein the instruction buffer is organized as a first-in first-out buffer.

25. (Currently amended) A microprocessor that defers execution of instructions with unresolved data dependencies as they are issued for execution in program order, comprising:

a microprocessor chip containing one or more microprocessor cores;

an execution mechanism within a microprocessor core configured to issue instructions for execution in program order during a normal execution mode;

a detection mechanism within the microprocessor core configured to detect an unresolved data dependency during execution of an instruction;

wherein if an unresolved data dependency is detected during execution of an instruction, the execution mechanism is configured to,

generate a checkpoint that can subsequently be used to return execution of the program to the point of the instruction, and to

execute subsequent instructions in an execute-ahead mode, wherein instructions that cannot be executed because of an unresolved data dependency are deferred, and wherein other non-deferred instructions are executed in program order;

wherein if a non-data-dependent stall condition is encountered while executing in normal mode or execute-ahead mode, the execution mechanism is configured to move into a scout mode;

wherein during scout mode instructions are speculatively executed to prefetch future loads, but results are not committed to the architectural state of the processor; and

wherein during scout mode when the launch point stall condition (the unresolved data dependency or the non-data dependent stall condition that originally caused the system to move out of normal execution mode) is finally resolved, the checkpoint is used to resume execution in normal mode from the launch point instruction (the instruction that originally encountered the launch point stall condition).